METHOD FOR EXECUTING A 32-BIT FLAT ADDRESS PROGRAM DURING A SYSTEM MANAGEMENT MODE INTERRUPT

CROSS-REFERENCE TO RELATED APPLICATION

[0001] This application claims the benefit of U.S. Provisional Patent Application No. 60/225,130, filed August 14, 2000, which is currently pending.

TECHNICAL FIELD

[0002] The described technology relates generally to a technique for executing a program during a system management mode interrupt.

BACKGROUND

[0003] The Intel Pentium processor (and other similar similar-class processors such as those from AMD, National Semiconductor, and ST Microelectronics) and its successors can run in three different operating modes: real mode, protected mode, and System Management Mode (SMM). (The architecture of the Pentium processor is described in "Pentium Processor User's Manual, Volume 3: Architecture and Programming Manual," dated 1993 and published by Intel Corporation, which is hereby incorporated by reference.) Real mode has been used for operating systems like DOS and provides a memory architecture that supports up to 1MB (megabyte) of address space. Protected mode is used by operating systems such as Linux and Windows NT, supporting a much larger address space of at least 4.2GB (gigabyte) and offering the capability of fetching 32-bit processor instructions from addresses beyond the 1MB address boundary. SMM is a special "invisible" mode of the processor that supports a limited amount (128KB, typically) of address space below 1MB for fetching 16-bit processor instructions. Additionally SMM can support data references throughout the entire

4.2GB address space. However, SMM is limited because it does not allow code to be executed when it is positioned above the 1MB boundary, making it impractical for use in running large or many applications. Further, those applications are limited to 16-bit operation, not flat 32-bit operation. Whether the processor is in real mode or protected mode, external circuitry can cause the processor to execute a special SMM Interrupt (SMI). When an SMI occurs, the operating system that is running is suspended and the SMI routine runs in a manner that is transparent to the operating system. This SMM was developed by the hardware manufacturers to enable power-management software to run transparently to the operating system. The SMM provides a small amount of RAM (128K typically) for use by the SMI routine. This memory is only mapped into an address space when in SMM. Also, while in SMM, the processor executes with real-mode addressing rules that are extended by making it possible to access memory beyond 1MB with addressing and size overrides on each instruction. In other words, the SMI routine does not run as flat 32-bit code, which means that 16-bit code needs to be used for the SMI routine.

BRIEF DESCRIPTION OF THE DRAWING

[0004] Figure 1 is a flow diagram illustrating the processing of the SMI routine's transition all the way to the 32-bit flat address mode environment.

DETAILED DESCRIPTION

[0005] A method and system for executing 32-bit flat address programs during a System Management Interrupt is provided. In one embodiment, the SMM technique provides a 16-bit SMI routine that is given control when an SMI occurs. That routine initially saves the state of the processor and then executes an instruction to switch to protected mode. When in protected mode, the routine transfers control to 32-bit code. The 32-bit code uses a global descriptor table ("GDT") that is different from that used by the interrupted operating system.

When the 32-bit code completes, it restores the saved processor state and returns from the interrupt by executing an RSM instruction.

[0006] In one embodiment, the SMI routine invokes a 32-bit operating system kernel that is developed using standard 32-bit Windows development tools. This kernel may be loaded into SMM memory by the BIOS during system initialization. SMIs are caused to occur periodically (e.g., every 16 milliseconds) by programming the power management timers on external control circuitry implemented in the "chipset" components on the motherboard. An SMI is generated when an SMI input line to the processor is set by the chipset. The 16-bit SMI routine loaded from the BIOS receives control when these SMIs occur, saves the processor state, switches into 32-bit flat protected mode, and transfers control to a 32-bit operating system kernel. The kernel runs entirely in 32-bit protected mode, and is capable of loading and running standard Windows NT Portable Executable programs within the SMI context. (Of course, these programs would typically only use APIs provided by the proprietary 32-bit kernel.) The 32-bit code executed during an SMI can run programs transparently to the foreground operating system and can run them even while the foreground operating system has crashed or stopped responding (e.g., Linux Panic, or Windows NT Blue Screen). This SMM technique enables various applications to be developed to execute during an SMI. The applications may include a remote console, remote boot, remote diagnostics, remote restart, and debugging.

[0007] Figure 1 is a flow diagram illustrating the processing of the SMI routine's transition all the way to the 32-bit flat address mode environment. In block 101, the routine saves the state of the processor, which may include the saving of the floating point register. In block 102, the routine switches to protected mode by initializing the Global Descriptor Table Register ("GDTR") to point to a new GDT, setting a bit in the processor's CR0 register to switch to protected mode, and finally loading segment registers with protected mode selectors mapped as 32-bit segments by the new GDT. In block 103, the routine transfers control to the 32-bit application code by a NEAR CALL instruction to the pre-loaded 32-bit kernel

dispatcher entrypoint. In block 104, the routine restores the saved state of the processor. In block 105, the routine returns from the interrupt by executing the RSM instruction.

[0008] In one embodiment, the BIOS would preload the 32-bit SMM kernel into SMRAM and/or other physical memory reserved for SMM activities early on during Power On Self Test (POST). The kernel would be retrieved from a compressed bit stream stored in the same Flash ROM as the BIOS, to minimize its impact on the overall size of the firmware footprint in the system. Any 32-bit flat address programs that would need to run in the environment would also be stored as compressed bit streams stored in the same Flash ROM or another Flash ROM in the system, and be subsequently loaded by the 32-bit kernel once it had received control in System Management Mode via an SMI.

[0009] In one embodiment, the SMM kernel would not have detailed knowledge of the programming of the chipset hardware (i.e., PIIX4 in the case of a Pentium II or Pentium III processor) in order to receive control via an SMI or dismiss the reason for the SMI at the hardware, since the mechanism for doing that is chipset dependent. The implementation would have this chipset programming (which both programs the chipset to generate SMIs on certain conditions and can dismiss those conditions at the hardware) handled by a Hardware Abstraction Layer (HAL), implemented outside the 32-bit SMM environment. This HAL would be implemented in two parts: the chipset programming itself, and the dispatching 16-bit SMIs to the 32-bit SMM kernel. In an implementation, the SMM manager component provides the dispatching functions, and the Chipset Personality Module (CSPM) provides the actual hardware programming functions. The SMM manager calls certain CSPM functions in order to perform SMI control programming on its behalf, making the details of the underlying hardware transparent to the SMM manager (and therefore the core BIOS) and the 32-bit SMM kernel and its applications.

[0010] Based on the above description it will be appreciated that, although various embodiments have been described for purposes of illustration, various

modifications may be made without deviating from the spirit and scope of the invention. Accordingly, the invention is not limited except as by the appended claims.